Amendments to the Claims

This listing of claims will replace all prior versions, and listings of claims in the application.

1. (currently amended) A method for data storage and retrieval from a network of servers, said method comprising the steps of:

a. defining an amount of n data pieces;

b. defining a minimal amount of data pieces k needed to restore a data file;

c. for a distributed arbitrarily-connected network of L servers, defining a number M of the servers that could be rendered inaccessible; and

d. creating M+k data pieces for storage on M+k servers;

whereby the ability to restore the data file from M servers is retained and the optimal utilization of data storage means obtained, and

wherein $k \leq n$ and $M+k<=L$.

2. (currently amended) A method for data storage and retrieval from a network of servers, said method comprising the steps of:

a. defining an amount of n data pieces;

b. defining a minimal amount of data pieces k needed to restore a data file;

c. for a distributed arbitrarily-connected network of L servers, defining a number M of the servers that could be rendered inaccessible; and

d. creating M+k data pieces for storage on M+k servers,

wherein said data pieces are numbered, interchangeable, and of equal size, and

wherein $M+k<=L$.

3. (canceled)

4. (original) The method as defined in claim 1 wherein M<L.

5. (currently amended) A method for data storage and retrieval from a network of servers, said method comprising the steps of:

a. defining an amount of n data pieces;

b. defining a minimal amount of data pieces k needed to restore a data file;

c. for a distributed arbitrarily-connected network of L servers, defining a number M of the servers that could be rendered inaccessible; and

d. creating M+k data pieces for storage on M+k servers,

wherein the number of data pieces M+k depends on the fault tolerance level of and the number of servers in the network, and

wherein M+k<=L.

6. (currently amended) A method for data storage and retrieval from a network of servers, said method comprising the steps of:

a. defining an amount of n data pieces;

b. defining a minimal amount of data pieces k needed to restore a data file;

c. for a distributed arbitrarily-connected network of L servers, defining a number M of the servers that could be rendered inaccessible; and

d. creating M+k data pieces for storage on M+k servers,

wherein the amount of redundancy data stored for each file is increased by an amount of about 1/k of the original file size, and

wherein M+k<=L.

7. (currently amended) A system for data storage and retrieval from a network of servers, said system comprising:

a predetermined amount of data pieces n;

a minimal amount of data pieces k needed to restore a data file;

a predetermined number M of servers in a network containing L servers, that could be rendered inaccessible; and

M+k data pieces for storage on M+k servers;

wherein the ability to restore a data file from M servers is retained and the optimal utilization of data storage means is obtained, and

wherein k≤n, and

wherein M+k<=L.

8. (currently amended) A system for data storage and retrieval from a network of servers, said system comprising:

a predetermined amount of data pieces n;

a minimal amount of data pieces k needed to restore a data file;

a predetermined number M of servers in a network containing L servers, that could be rendered inaccessible; and

M+k data pieces for storage on M+k servers,

wherein said data ~~bases~~ pieces are numbered, interchangeable, and of equal size, and

wherein M+k<=L.

9. (canceled)

10. (original) The system as defined in claim 7 wherein M<L.

11. (currently amended) A system for data storage and retrieval from a network of servers, said system comprising:

a predetermined amount of data pieces n;

a minimal amount of data pieces k needed to restore a data file;

a predetermined number M of servers in a network containing L servers, that could be rendered inaccessible; and

M+k data pieces for storage on M+k servers,

wherein the number of data pieces M+k depends upon the fault tolerance level and the number of servers in the network, and

wherein M+k<=L.

12. (currently amended) A system for data storage and retrieval from a network of servers, said system comprising:

a predetermined amount of data pieces n;

a minimal amount of data pieces k needed to restore a data file;

a predetermined number M of servers in a network containing L servers, that could be rendered inaccessible; and

M+k data pieces for storage on M+k servers,

wherein the amount of redundancy data stored for each file is increased by an amount of about 1/k of the original file size and can vary for each file, and

wherein M+k<=L.

13. (previously presented) The method of claim 1, further comprising creating at least M+k data pieces for storage on at least M+k servers.

14. (previously presented) The method of claim 1, wherein the same algorithm is used to create the M+k data pieces.

15. (previously presented) The method of claim 1, wherein the number L is variable.

16. (previously presented) The system of claim 7, further comprising at least M+k data pieces for storage on at least M+k servers.

17. (previously presented) The system of claim 7, wherein the same algorithm is used to create the M+k data pieces.

18. (previously presented) The system of claim 7, wherein the number L is variable.

19. (currently amended) A method for data storage comprising:

defining a plurality of n data units that correspond to a data file;

defining a minimal number k of data units required to restore the data file;

defining a number of M servers out of a plurality of L servers that could be rendered inaccessible; and

creating M+k functionally equivalent data units for storage on M+k servers out of the plurality of L servers,

wherein $M+k<=L$.

20. (previously presented) The method of claim 19, wherein the data units are numbered and of equal size.

21. (previously presented) The method of claim 19, wherein $k \leq n$.

22. (previously presented) The method of claim 19, wherein the number M+k depends on a fault tolerance level of a network formed by the plurality of L servers.

23. (canceled)

24. (previously presented) The method of claim 19, wherein the number M+k is dynamically adjustable based on a fault tolerance level of a network formed by the plurality of L servers.

25. (previously presented) The method of claim 19, wherein the plurality of L servers form a distributed arbitrarily-connected network.

26. (previously presented) The method of claim 19, wherein the amount of redundancy data stored for each data file is increased by an amount of about 1/k of the original data file size.

27. (previously presented) The method of claim 19, further comprising creating at least M+k data units for storage on at least M+k servers.

28. (previously presented) The method of claim 19, wherein the same algorithm is used to create the M+k data units.

29. (previously presented) The method of claim 19, wherein the number L is variable.

30. (currently amended) A method for data storage comprising:

defining a plurality of n data units that correspond to a data file;

defining a number k of data units required for restoring the data file;

defining a number of M servers out of a network of L servers that could be rendered inaccessible, wherein the number M is dynamically adjustable based on a fault tolerance level of the network; and

creating M+k data units for storage on M+k servers,

<u>wherein M+k<=L</u>.

31. (previously presented) The method of claim 30, wherein the data units are numbered and are of equal size.

32. (previously presented) The method of claim 30, wherein $k \leq n$.

33. (previously presented) The method of claim 30, wherein the number M+k depends on the number L.

34. (canceled)

35. (previously presented) The method of claim 30, wherein all the data units are functionally equivalent.

36. (previously presented) The method of claim 30, wherein the amount of redundancy data stored for each file is increased by an amount of about 1/k of the original data file size.

37. (previously presented) The method of claim 30, wherein the network of L servers is a distributed arbitrarily-connected network.

38. (previously presented) The method of claim 30, further comprising creating at least M+k data units for storage on at least M+k servers.

39. (previously presented) The method of claim 30, wherein the same algorithm is used to create the M+k data units.

40. (previously presented) The method of claim 30, wherein the number L is variable.

41. (currently amended) A system for data storage comprising:

n data units corresponding to a data file;

k data units required to restore the data file;

M servers in a network of L servers that could be rendered inaccessible; and

M+k functionally equivalent data units for storage on M+k servers out of the L servers,

<u>wherein M+k<=L</u>.

42. (previously presented) The system of claim 41, wherein the data units are numbered and are of equal size.

43. (previously presented) The system of claim 41, wherein $k \leq n$.

44. (previously presented) The system of claim 41, wherein the number M+k depends on a fault tolerance level of the network.

45. (canceled)

46. (previously presented) The system of claim 41, further comprising at least M+k data units for storage on at least M+k servers.

47. (previously presented) The system of claim 41, wherein the same algorithm is used to create the M+k data units.

48. (previously presented) The system of claim 41, wherein the number L is variable.

49. (previously presented) The system of claim 41, wherein the amount of redundancy data stored for each data file is increased by an amount of about 1/k of original data file size and can vary for the each data file.

50. (previously presented) The system of claim 41, wherein the network of L servers is a distributed arbitrarily-connected network.

51. (currently amended) A system for data storage comprising:

n data units corresponding to a data file;

k data units required to restore the data file;

M servers in a network of L servers that could be rendered inaccessible, wherein the number M is dynamically adjustable based on a fault tolerance level of the network; and

M+k data units for storage on M+k servers out of the L servers,

wherein M+k<=L.

52. (previously presented) The system of claim 51, wherein the data units are numbered and of equal size.

53. (previously presented) The system of claim 51, wherein all the data units are functionally equivalent.

54. (previously presented) The system of claim 51, wherein $k \leq n$.

55. (previously presented) The system of claim 51, wherein the number M+k depends upon the number L.

56. (canceled)

57. (previously presented) The system of claim 51, wherein the amount of redundancy data stored for each data file is increased by an amount of about 1/k of original data file size and can vary for each data file.

58. (previously presented) The method of claim 51, wherein the network of L servers is a distributed arbitrarily-connected network.

59. (previously presented) The system of claim 51, further comprising at least M+k data units for storage on at least M+k servers.

60. (previously presented) The system of claim 51, wherein the same algorithm is used to create the M+k data units.

61. (previously presented) The system of claim 51, wherein the number L is variable.

62. (currently amended) A computer program product for data storage, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor, the computer program logic comprising:

computer program code means for defining a plurality of n data units that correspond to a data file;

computer program code means for defining a minimal number k of data units required to restore the data file;

computer program code means for defining a number of M servers out of a plurality of L servers that could be rendered inaccessible; and

computer program code means for creating M+k functionally equivalent data units for storage on M+k servers out of the plurality of L servers,

wherein M+k<=L.

63. (previously presented) The computer program product of claim 62, wherein k≤n.

64. (previously presented) The computer program product of claim 62, wherein the number M+k depends on a fault tolerance level of a network formed by the plurality of L servers L.

65. (canceled)

66. (previously presented) The computer program product of claim 62, wherein the number M+k is dynamically adjustable based on a fault tolerance level of a network formed by the plurality of L servers.

67. (previously presented) The computer program product of claim 62, wherein the plurality of L servers form a distributed arbitrarily-connected network.

68. (previously presented) The computer program product of claim 62, wherein the amount of redundancy data stored for each data file is increased by an amount of about 1/k of the original data file size.

69. (previously presented) The computer program product of claim 62, further comprising at least M+k data units for storage on at least M+k servers.

70. (previously presented) The computer program product of claim 62, wherein the same algorithm is used to create the M+k data units.

71. (previously presented) The computer program product of claim 62, wherein the number L is variable.

72. (currently amended) A computer program product for data storage, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor, the computer program logic comprising:

computer program code means for defining a plurality of n data units that correspond to a data file;

computer program code means for defining a number k of data units required for restoring the data file;

computer program code means for defining a number of M servers out of a network of L servers that could be rendered inaccessible, wherein the number M is dynamically adjustable based on a fault tolerance level of the network; and

computer program code means for creating M+k data units for storage on M+k servers,

<u>wherein M+k<=L</u>.

73. (previously presented) The computer program product of claim 72, wherein the data units are numbered and are of equal size.

74. (previously presented) The computer program product of claim 72, wherein $k \leq n$.

75. (previously presented) The computer program product of claim 72, wherein the number M+k depends on the number L.

76. (canceled)

77. (previously presented) The computer program product of claim 72, wherein all the data units are functionally equivalent.

78. (previously presented) The computer program product of claim 72, wherein the amount of redundancy data stored for each file is increased by an amount of about 1/k of the original data file size.

79. (previously presented) The computer program product of claim 72, wherein the network of L servers is a distributed arbitrarily-connected network.

80. (previously presented) The computer program product of claim 72, further comprising at least M+k data units for storage on at least M+k servers.

81. (previously presented) The computer program product of claim 72, wherein the same algorithm is used to create the M+k data units.

82. (previously presented) The computer program product of claim 72, wherein the number L is variable.